Information Retrieval of Mass Encrypted Data over Multimedia Networking with N-Level Vector Model-Based Relevancy Ranking

Jinghui Peng, Shanyu Tang*, *Senior Member*, *IEEE*, Liping Zhang, Ran Liu

Abstract

With an explosive growth in the deployment of networked applications over the Internet, searching the encrypted information that the user needs becomes increasingly important. However, the information search precision is quite low when using Vector space model for mass information retrieval, because long documents having poor similarity values are poorly represented in the vector space model and the order in which the terms appear in the document is lost in the vector space representation with intuitive weighting. To address the problems, this study proposed an N-level vector model (NVM)-based relevancy ranking scheme with an introduction of a new formula of the term weighting, taking into account the location of the feature term in the document to describe the content of the document properly, investigated into ways of ranking the encrypted documents using the proposed scheme, and conducted realistic simulation of information retrieval of mass encrypted data over multimedia networking. Results indicated that the timing of the index building, the most costing part of the relevancy ranking scheme, increased with the increase in both the document size and the multimedia content of the document being searched, which is in agreement with the expected. Performance evaluation demonstrated that our specially designed NVM-based encrypted information retrieval system is effective in ranking the encrypted documents transmitted over multimedia networks with large recall ratio and

great retrieval precision.

Key Words: encrypted data retrieval; N-level vector model; relevancy ranking; multimedia security

1. Introduction

Many data transmissions over networks are based on the multimedia network transport protocol. The sensitive multimedia data transmitted over the Internet are often encrypted to protect them from interception. It is imperative that the Internet users are able to search the encrypted information they need. As cloud computing is widely used in massive data storage, ranking the encrypted documents in a cloud environment is one of the critical problems in the information security field.

The Internet users benefit from having a variety of online services provided by the Internet providers. On the one hand, in order to save the resource needed in data storage, management and computing, and lower the cost, the service providers gradually transfer from offering client software and hardware to providing specific services. Storage and computing resources have the trend of aggregating, and more and more data are being stored on remote servers.

On the other hand, cloud computing is the latest and prevalent idea that provides storage and computation services. Cloud computing refers to both the applications delivered as services over the Internet, and the hardware and system software in the datacenters that provide those services [1]. The services themselves have long been referred to as Software (Service), so the datacenter hardware and software are called a Cloud. Major IT giants have launched their own cloud computing products. Typical applications of cloud computing include various Internet services, which may be effective solutions to data storage and management.

When the data storage server is not trustable, there is a risk of leaks. First, when users storage their

information on the server, public information may be collaboratively used to infer personal information, thus sensitive public information should be protected. Second, there are needs of multiple users information management in enterprise user groups. And in such systems, the data of enterprise user groups should not be accessible to all the users in the group. Although access control technology can solve the problem to some extent, it could not prevent privacy violation or the leaking of information to the server, and data protection remains to be done.

In the same way, for cloud computing and cloud storage, the service provider can also catch information of users. Thus, the security issue of cloud storage becomes prominent. In short, with the enlarging of a corpus of data outsourced to storage servers or cloud servers, the security issue becomes the coming problem [2]. It is a fundamental approach to protect the data by encrypting them before being outsourced to servers [3].

However, along with the growing size of the encrypted information comes the problem of searching over the cipher. The existing encryption technology is very mature to ensure secure data storage, and information retrieval technology has also made a great progress, but encrypted information retrieval technology is still in the relatively last start of the study, especially information retrieval technology in multimedia cipher data. Although the data stored can be encrypted easily, retrieving the specific information from the mass cipher data is very difficult.

With the enlarging of encrypted data, the number of possible relevant documents may be very large with the expansion of the collection. Consequently it is important to provide the most relevant documents to users to better meet their information demands by ranking the encrypted documents. Nevertheless, the traditional information retrieval method based on Boolean model does not take into account the user behaviour and the keyword access frequency. The method is only efficient in telling what documents should be retrieved, and it is not likely to tell the user which documents are more relevant if there are lots of documents retrieved. Thus, the

out of order ranking problem often occurs, which makes it hard for the data consumers to find the subsets which are most likely to satisfy their requirements.

Vector space model is widely used in the field of information retrieval for its simplification and adaptability. And it addresses the out of order ranking problem associated with the Boolean model-based information retrieval method. However, the information search precision is relatively low when using the vector space model for information retrieval, since long documents are poorly represented in the vector space model because they have poor similarity values, and weighting is intuitive but not very formal in the vector space model. In addition, the order in which the terms appear in the document is lost in the vector space representation.

To address the problems above, this research endeavoured to propose an N-level vector model-based relevancy ranking scheme with an introduction of a new formula of the term weighting, taking into account the location of the feature term in the document so as to describe the content of the document properly, investigated into how to rank the encrypted documents using the proposed scheme, and studied the theory and applications of information retrieval of mass encrypted data over multimedia networking.

This rest of the paper is organized as follows. In Section 2, the related information retrieval methods are reviewed. Followed by Section 3 in which the proposed N-Level vector model-based relevancy ranking scheme is detailed. The experiments on searching engine based on the proposed N-level vector model as well as results discussion are given in Section 4. Finally, a conclusion is drawn in Section 5.

2. Related Work

To guarantee the security while searching over encrypted information, a variety of secure search schemes over encrypted information have been proposed. Search schemes over the cipher text include single-user linear search, a searchable keyword-based encryption and security index, and some other search methods.

With regard to single-user linear search, Goldreich and Ostrovsky proposed Oblivious RAM scheme, which could retrieve encrypted data when the query feature is in hiding [4]. However, this algorithm requires polynomial time for interaction, and the server needs exponential polynomial computations. Based on Ostrovsky and Skeith's scheme, Yerukhimovich suggested homomorphic public key encryption-based information retrieval, which analysed communication and computational complexity [5]. In 2000, Song et al introduced a practical search method while guaranteeing security, and this method requires only one interaction. However, there is a linear relationship between the calculation and the number of data [6].

In the area of searchable keyword-based encryption, Boneh proposed a searchable asymmetric public key encryption scheme with the same essence as Song's scheme [7]. Although the schemes are provably secure, they are extremely time consuming and are intolerable when the encrypted information is huge, such as in a cloud environment. With the increasing of the size of the encrypted information, it is inefficient to perform linear search. Golle et al suggested an algorithm that allows multi-search on the server, and the data of which were generated by a trustable third party with public key [8]. And Abdalla et al put forward a public key encryption algorithm to ensure that the user's query was not informed of the server [9]. This algorithm used Bloom filter to achieve space saving so as to achieve public-key encryption.

As to security index, Goh firstly proposed a security indexing method based on Bloom filter and it is essentially a binary vector and random mapping function. Chang et al improved the security on the basis of Goh's algorithm, and they suggested the introduction of a semi-trusted third party to coordinate and mediate queries in multi-server [10]. Park et al developed the security indexing algorithm, and a multi-user information sharing system with the featured employment of Bloom filter following Goh's work on secure index [11-12]. In their algorithms, the encryption keys are firstly used to generate the inverse hash sequence in each encryption, and the encrypted index is then put into the bloom filters. The inverse hash sequence is used to build a trap-door

when retrieving, and then for bloom detection. Finally, the returned cipher texts are decrypted to get the needed documents.

Bloom filter is a high space efficient data structure that can swiftly check if an element exists in a set. This system could guarantee the security of initial searches. Though Bloom filter is faster in conducting one by one linear search, it is also inefficient to perform searching when the size of the encrypted information is huge. Thus, it is imperative to follow the general information retrieval by building an inverted index. This method can solve the problem subject to statistical attacks, but it needs to generate lots of key sequences causing the computational complexity to increase linearly with the data retrieval time. Therefore, this method is not suitable for practical application in the massive network data retrieval.

Nevertheless, the methods mentioned above can only indicate which documents should be retrieved, but cannot tell which documents are more relevant if there are a mass of documents retrieved. So they cannot be directly used for mass encrypted information retrieval over multimedia networks. Hence, the problem of ranking over the encrypted documents becomes prominent and urgent. As the size of information is growing unprecedentedly, the demands of improving the retrieval recall and the retrieval precision by ranking are rising. Ranking the possible documents according to relevance would better meet user's information needs. Motivated by a need to achieve a reliable and effective information retrieval of mass encrypted data over multimedia networking, we proposed an N-level vector model-based relevancy ranking scheme in this study, and the proposed scheme is described in detail in the next section.

3. N-Level Vector Model-Based Relevancy Ranking

In this section, we first analyse the limitations of the existing vector space model in dealing with Chinese information retrieval. Subsequently, we describe the proposed N-level vector model-based relevancy raking

scheme for Chinese information retrieval over multimedia networking, detailing the scheme in terms of representation of text, selection and weighting of the term, similarity and relevancy ranking.

3.1 Limitations of existing vector space model

Vector Space Model is widely used in the field of information retrieval, in which documents and queries are both represented as vectors. And it is based on the comparison of the query term vector with the document term vectors. It turns information retrieval into a matter of matching the vectors, and the content of the document becomes a vector of index term and its weight.

Vector space model is an algebraic model which selects the index item from the text, and weights the item selected in some way [13-15]. In the vector space model, documents and queries are both represented as vectors, and the vectors are then compared and matched. The distance between vectors indicates the similarity between documents.

Vector space model is widely used in modern information retrieval for its simplification and adaptability. However, the traditional term frequency-inverse document frequency (*tf-idf*) method used obtains a poor result in information retrieval, because it cannot describe the content of the documents properly, and the search precision is relatively low when using the vector space model for information retrieval [16].

In addition, the vector space model has three main limitations: the order in which the terms appear in the document is lost in the vector space representation; long documents are poorly represented because they have poor similarity values (a small scalar product and a large dimensionality); weighting is intuitive but not very formal.

3.2 Representation of Chinese text

For a Chinese document, the text consists mainly of some basic symbols of language such as characters and punctuations. Representing the document in a simple and accurate way is the prerequisite of information

retrieval.

To represent a Chinese document, an index item is selected from the text, and the selected item is weighted in some way. The document is represented as a vector, given by:

$$D(t_1, t_2, ..., t_k, ..., t_n) \quad (1)$$

where t_k is a term of the document *D*, and $1 \le k \le n$. Considering each document as a vector, the document *D* contains *n* index terms. Since the terms weight differently, their weights are denoted as $w_1, w_2, ..., w_n$. The document *D* is then represented as:

$$D(t_1, w_1; t_2, w_2; ..., t_k, w_k; ..., t_n, w_n) \quad (2)$$

where the weight w_k corresponds to the term t_k, and $1 \le k \le n$, as shown in Fig. 1.

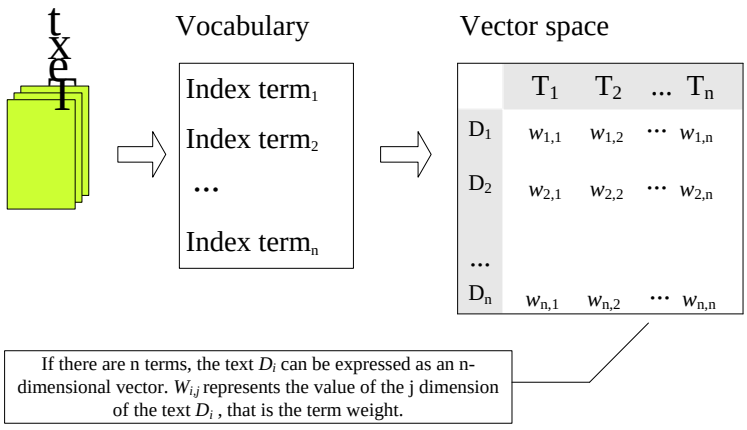

Fig. 1 Representation of text in vector space

3.3 Selection and weighting of the index term

3.3.1 Selection of the index term

The index terms include a variety of different types. Typically terms are single words, keywords, or longer phrases. The selection of the index term depends on the processing speed, storage space and processing accuracy [14]. There are some principles for the selection. Firstly, the term selected should be a language unit

which contains enough information and does well in expressing texts. Secondly, the probability of the text should be subject to a relatively significant statistical law to ensure that it is applicable in the document classification, information retrieval, and other applications. Thirdly, it should be easy to select the index term with lower complexity of time and space. Practically, words and phrases are often used as feature items.

Since Chinese is an ideogram, word segmentation should be done firstly, the Chinese document is then divided into notional and function words, and finally the feature terms are selected.

There are two methods to select terms. The first one is linguistic analysis, taking exclusion. Some auxiliary, function words and common words are collected to establish a vocabulary. The other one is mathematical statistics. The notional and function words are distinguished according to the differences between their frequencies. And it has shown that the effectiveness of the simple method of selection is as well as the complex one.

3.3.2 Weighting of the index term

Each term has a certain weight which reflects its descriptiveness with respect to the query or document. The weight of terms reflects the importance of them. Distinguishing texts to the greatest extent is the principle of weighting the terms. To ensure the retrieval recall and precision, it should contain the factor of recall and the factor of precision at the same time while weighting the terms. The factor of weighting consists of the normalization factor, term frequency, and inverse document frequency [15].

Several different ways of computing the values of terms have been developed. Since terms which appear more frequently weight higher, the term frequency (*tf*) is typically used to weight the term, and the retrieval recall can be improved through querying by terms with higher appearing frequency.

The term frequency can not guarantee good retrieval performance alone. In information retrieval, the recall and precision are regarded as two inter-constraint measures. Thus, the factor of document frequency needs to be

added in order to increase the partition degree of documents. The less the term appears in documents, the greater the inverse document frequency (*idf*) is. When the total number of documents is *N*, in which the number of documents containing the term is *n*, the inverse document frequency is defined as $idf = \log(N/n)$.

Good query vectors contain feature terms which can distinguish specific documents with other documents. And these terms should have a relatively high frequency, but appear in few documents in the collection of documents. So the definition of term frequency and inverse document frequency are combined to produce a composite weight for each term in each document. The model is known as term frequency-inverse document frequency model.

The weight for the term is *w*, given by

$$w_{ik} = \frac{tf_{ik}}{df_k} = tf_{ik} \times idf_k \tag{3}$$

where tf_{ik} is the term frequency of term t_k in document D_i, df_k is the document frequency which is the number of documents containing the term t_k in the collection of documents *D*, and idf_k is the reciprocal value of df_k, namely the inverted document frequency. In addition, the length of document should be taken into account. If not considering this, a long document could be easily detected, whereas a short document could be missed. So the normalization treatment on formula is also necessary.

3.4 Similarity

Similarity is used to represent the degree of correlation between two documents. Both documents and queries can be represented as vectors, which means that information retrieval turns into a matter of matching the vectors. The distance between vectors indicates the similarity between documents. And the process of information retrieval is to calculate the similarity between document and query, then rank the search results according to similarity. Using the inner product range between vectors (D_1, D_2), the similarity *sim* can be calculated as [16]:

$$sim(D_1, D_2) = \sum_{k=1}^{n} w_{1k}\, w_{2k} \qquad (4)$$

where *w* is the term weight, or using the cosine, the similarity between vectors can be calculated as:

$$sim(D_1, D_2) = \cos\theta = \frac{\sum_{k=1}^{n} w_{1k}\, w_{2k}}{\sqrt{(\sum_{k=1}^{n} w_{1k}^2)(\sum_{k=1}^{n} w_{2k}^2)}} \qquad (5)$$

The relevance of the document *D* to the query *Q*, denoted by $sim(\bar{q}, \bar{d})$, is then evaluated by using a measure of similarity between vectors. The query of the user is represented as a vector *Q*, where q_j is the weight of term T_j in the query. All documents and queries can be mapped to the vector space *T*, which turns information retrieval into the issues of matching the vectors (Fig. 1). The similarity between queries and documents retrieved can be calculated using the inner product range or cosine, given by:

$$sim(\bar{q}, \bar{d}) = \cos\theta = \frac{\sum_{i=1}^{n} q_i \times d_i}{\sqrt{(\sum_{i=1}^{n} q_i^2)} \times \sqrt{(\sum_{i=1}^{n} d_i^2)}} = \frac{\sum_{i=1}^{n} q_i \times w_{ij}}{\sqrt{(\sum_{i=1}^{n} q_i^2)} \times \sqrt{(\sum_{i=1}^{n} w_{ij}^2)}} \qquad (6)$$

Similarity ranges from 0 to 1, namely, when the angle between the vectors is 0, the similarity has the maximum value 1, which means the document is most relevant to the query.

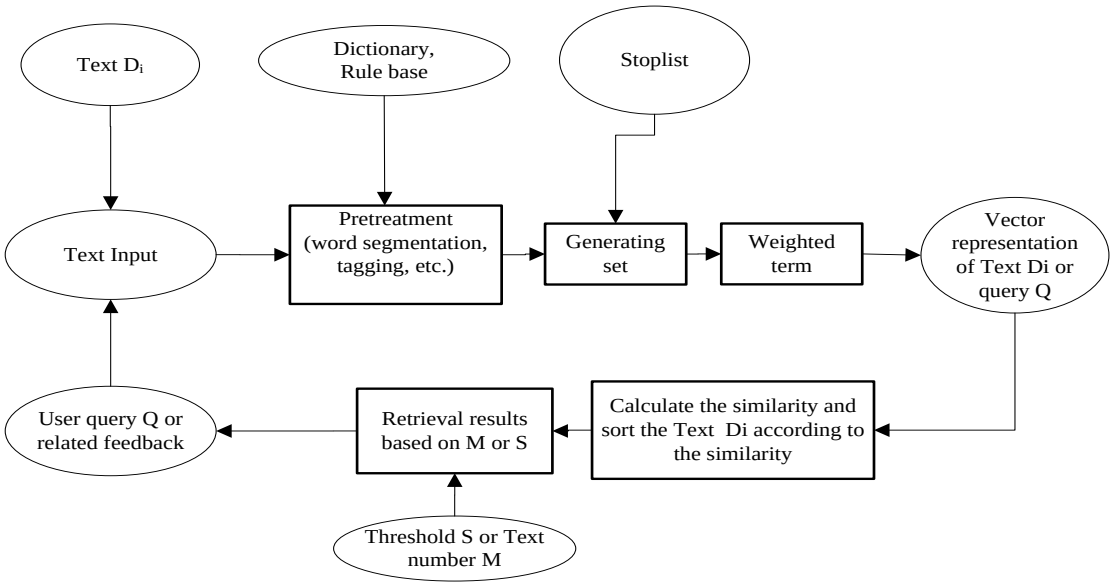

Fig. 2 Selection and weighting of the index term and similarity calculation

As shown in Fig. 2, the similarity measure is to compare the query vector and the document vector set, and to return the documents with a higher correlation, so that documents are sorted by relevance. An optional threshold can also be offered to adjust the number of documents returned.

3.5 N-level vector model-based relevancy ranking

We propose an N-level vector model (NVM) that divides the text into *N* relatively independent text segments according to the importance of the content, and then establishes the text feature vector and the text weight vector according to the contents of the text segments. Compared with the traditional vector space model, the proposed model can reflect the characteristics of the document better, and can describe the document properly [17]. In addition, the calculation methods of the feature vector and the similarity are defined much more precisely such that the corresponding algorithms can adapt the dynamitic extension of the document set. The N-level vector model has higher precision and faster computation speed than the vector space model.

In the proposed N-level vector model, the text segment refers to a text portion with a different role in the document D_i. And title, abstract, text and so on can be regarded as the text segments. The segment of the *j* part/section of the document is denoted as S_{ij}.

The weight of the feature term refers to the relative importance that the feature t_k represents in the text segment S_i, which is denoted as W_{ik}. W_{ik} is based on the ratio of tf_{ik} and l_i (the size of the text segment S_i), and can be calculated as:

$$W_{ik} = tf_{ik} / l_i \tag{7}$$

In the process of information extraction and query matching, the same term at different locations in the document performs differently in expressing the content of the document. The existing vector space model using

tf-idf weighting to perform the calculation of the weight of terms, but the weight of terms depends entirely on the appearing frequency and does not take the term position into account [18]. However, the document in reality is often clearly divided into a number of parts, such as title, abstract, keywords, text, etc. And the words in the title is obviously important than the words in the body by artificial judgment, so the vector space model cannot distinguish them. To solve this problem, we in this study introduce a new formula of term weighting above. And in the calculation of term weighting, different standards are used in different parts of the document [19]. The terms of each segment are multiplied by different scale factors, thereby improving the similarity calculation significantly, and getting better retrieval results.

The text term vector is a sequence of all the feature items that appear in the text segment S_i according to the corresponding weight of the feature item, which is denoted as (t_1, t_2, …, t_k), where *n* refers to the number of features. The text weight vector is a vector of corresponding weights of items, which is denoted as (W_{i1}, W_{i2}, …, W_{ik}).

The N-level vector model is suggested with the above definition in this study. With the proposed model, a document is divided into *n* layers according to the organizational structure, and the text feature vector and the text weight vector are then established according to the contents of the text segments [13].

The query submitted by the user is denoted as *QS*. Before the similarity comparison with the text segment $S_i(t_1, t_2, \ldots, t_k)$, the weight of query vector is calculated by Boolean model [20]:

$$q_j = \begin{cases} 1, & t_j \in QS \\ 0, & t_j \notin QS \end{cases} \tag{8}$$

where $j \in [1,2,...,k]$.

Query text is represented as a space vector $QS(q_1, q_2, ..., q_k)$. The similarity between the document segment S_i and the query text is obtained:

$$sim(QS, S_i) = \cos\theta_i = \frac{\sum_{j=1}^{k} W_{ij} \times q_j}{\sqrt{\left(\sum_{j=1}^{k} W_{ij}^2\right)\left(\sum_{j=1}^{k} q_j^2\right)}} \qquad (9)$$

where $i \in [1,2,...,N]$.

N in the above formula refers to the number of layers the document is divided into. After calculating the similarity between the query text *QS* and the segments of the document, the similarity between document d_l and query *QS* is obtained:

$$sim(QS, d_l) = \sum_{j=1}^{N} sim(QS, S_{li}) / N = \sum_{j=1}^{N} \cos\theta_j / N \qquad (10)$$

The similarity calculated using the formula above was used to rank the documents retrieved over simulated multimedia networking detailed in the next section.

3.6 Advantages of using the proposed method

The traditional vector space model uses *tf-idf* weighting to compute the weight of terms that depends entirely on the appearing frequency of the terms, but the terms appear in title, abstract, keywords, text, etc. have different levels of importance (relevance), which means that the vector space model does not take into account the term location in the document being searched. The proposed N-level vector model-based relevancy ranking scheme divides the text into N relatively independent text segments according to the importance of the content, computed using a new formula of the term weighting with the location of the feature term included, and then establishes the text feature vector and the text weight vector according to the contents of the text segments. So the proposed scheme reflects the characteristics of the document better, and describes the document properly compared with the traditional vector space model.

In short, the proposed information retrieval method introduces a new formula of the term weighting, taking into account the location of the feature term in the document to describe the contents of the document properly, and improving the similarity calculation significantly, thereby getting better retrieval results.

4. Experimental Results and Discussion

4.1 NVM-based encrypted information retrieval system

An N-level vector model-based encrypted information retrieval system was designed and implemented in this study. The information retrieval system was composed of three components, the index module to generate index, the search module used to search encrypted information, and the user interface module in which the user performs information retrieval through the input and output interfaces provided.

The NVM-based encrypted information retrieval system is illustrated in Fig. 3. The information retrieval system consists of User interface, Search engine, Index database, and NVM-based ranking. The user interface enables the user to submit queries and provides presentation in response to user queries. The search engine discovers, crawls, transforms and stores documents for retrieval on the index database in which text extraction, text analysis, and index establishment are conducted. The encrypted information retrieval is realised according to NVM-based ranking scores computed using equation (10) detailed in the previous section. The main stages involved are query parsing, lexical analysis, language processing, searching index, and ranking the retrieval results.

4.1.1 Index module

The index module was to establish the index of the information collected from the collector and stored on the index database for the query purpose. The information acquisition was analyzed and the resulting documents were tabulated as the page-to-page correlations. The main aims of the index were to reorganize the description of these documents, and build the inverted list of the important data items, ready for the user's search.

The index module was based on N-level vector model-based relevancy ranking. The indexing process was divided into three main stages: text extraction, text analysis, and index establishment.

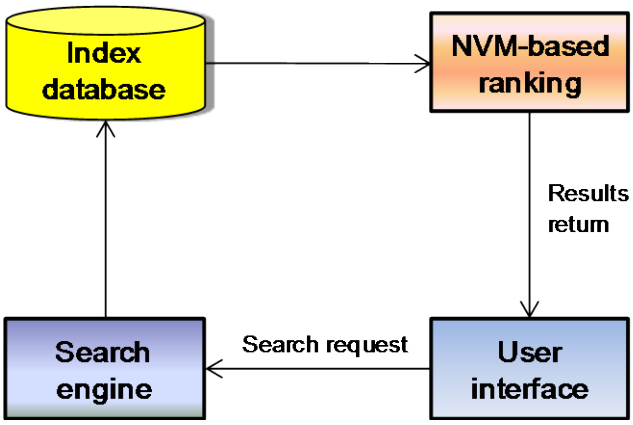

Fig. 3 Illustration of NVM-based encrypted information retrieval system

The vector analysis of document keywords was to allocate the set of keywords to a dictionary, establish a vector value for each document according to the position of the corresponding keyword in the dictionary, and then assign the vector value as the document keyword index.

In the index building block, the first step was to seek all the words from the document string, that is, word segmentation. The words in English are separated by spaces, which are easier to deal with. But Chinese words are connected together, so it needs to process word segmentation first. Normally, a Chinese document contains some function words used for connection purposes only, and those words do not represent any meaningful concept which can be filtered out. User queries often need to be unified in all words, and usually need to be reduced to different tenses, voice. Punctuation marks in the article are often not indicative of a concept, so they can be filtered out. In this module, these measures were conducted by the *Analyzer* Java class, as shown below:

```
Directory dir = FSDirectory.open(Paths.get(indexPath));
Analyzer analyzer = new StandardAnalyzer();
IndexWriterConfig iwc = new IndexWriterConfig(analyzer);
```

With the keywords being extracted, a single row index was set up. The corresponding relationship of the inverted index was that the keyword mapped to all the documents containing the keyword. Obviously it was not enough to know the keyword appeared in what documents, and it should know the number and location of the

keyword appeared in the documents. The appearing location included the character position and the keyword position. The character position was used to record the character order in which the keyword appeared in the document, and the advantage of the character position was fast positioning with the keyword being explicitly highlighted. The keyword position was to record the order in which the keyword appeared in the document, and its advantages were quick phrase queries with less index space costing [21].

A binary search algorithm [22] was used to achieve fast positioning of keywords in this study. The function was realized by the *IndexFiles* Java class, as shown below:

```
public class IndexFiles {

    private IndexFiles() {
    }

    /** Index all text files under a directory. */
    static void index_Files() {
```

4.1.2 Search module

The search module was used to search the corresponding relevant documents from the index database according to the user query, evaluate the correlation / relevancy of the query and the documents, and then return a set of documents whose relevancy degrees were consistent with a particular threshold [23]. The retrieval schemes included specific concept-based retrieval, keyword-based retrieval, and content-based retrieval.

As shown in Fig. 4, having completed the document content index, the search module provided users with retrieval services. The module first accepted the user search request, accessed the corresponding index database according to the request, and then ranked the set of search results according to the relevancy degrees before returning to the user.

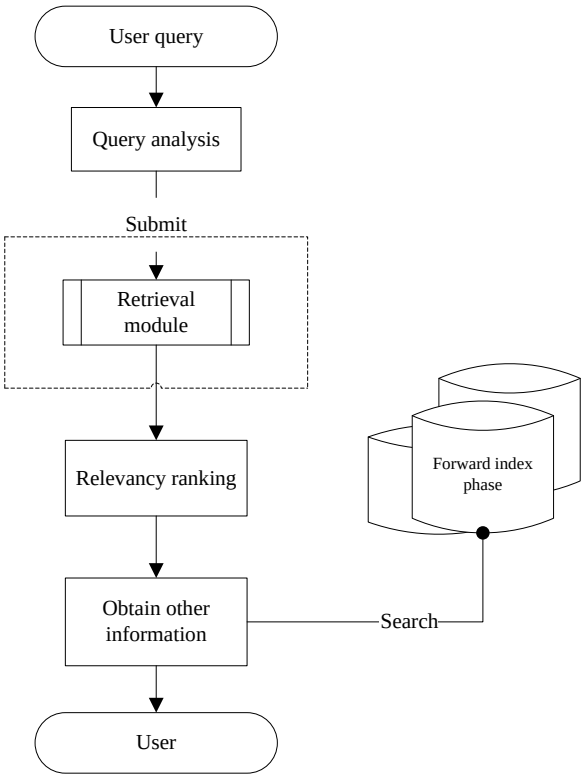

Fig. 4 Flow chart of information retrieval process

Through the input interface, the user submitted a search request to the search engine, which searched the local index library upon receiving the user query. When the relevant documents were found, they were ranked in accordance with the algorithm, and returned to the user.

In search of the index, a binary search was used to find the keyword from the dictionary, read out all the documents by pointing to the pointers of the frequency files, and then return the results. The dictionary was generally small, so the time costing of the whole process was within a millisecond.

Retrieval was mainly divided into four steps. The first step was the user entering the query language; the second step included query parsing, lexical analysis, and language processing; the third step was searching index, and then locating the documents that met the requirements of the grammar; the fourth step was ranking the retrieval results according to the correlation between the resulting documents and the queries. These functions were implemented by the *SearchFiles* Java class, as shown below:

```
public class SearchFiles {

    private SearchFiles() {
    }

    /** Simple command-line based search demo. */
    @SuppressWarnings({ "unused", "static-access" })
    static void search_Files() throws Exception {
```

4.1.3 User interface module

The user interface was to provide a visual query input and result output interface. In the result output interface, the search engine presented the searched results in a linear list of documents, containing the document location, the document title, the text, and other information. The user needed to browse to seek the required documents.

```
class ActionHandler implements ActionListener {
    public void actionPerformed(ActionEvent e) {

 if (e.getActionCommand() == "Index") {
     // new blankWindow().source_directory.setText("Index");
     IndexFiles.index_Files();

  if (e.getActionCommand() == "Search") {
      // new blankWindow().index_directory.setText("Search");
      try {
          SearchFiles.search_Files();
```

The search engine provided the user's retrieval interface with the *blankWindow* Java class, mainly through calling *IndexFiles* and *SearchFiles* to achieve the index construction and search functions. There were Construction window, Input window, and Search and Index buttons to the user. In the input window the user entered the document library to be searched, and the specified index storage path. When no index storage path was specified, the default path would be the search engine path. The output box displayed the time used to generate the index. The number of the documents matched to the keyword and the documents searched were displayed by clicking on the Search button in the interface.

4.2 Results and discussion

In our experiments, the NVM-based encrypted information retrieval system was implemented on a computer having a processor speed of Intel(R) Pentium(R) CPU G2020 @ 2.90 GHz, 4.00 GB of RAM and a

hard disk capacity of 465 GB, with an instant access to the Internet throughout a local area network.

Java Development Kit (JDK) was installed on the computer, including the Java runtime environment, as well as the Java class libraries, and Java tools, which are the basis of Java operation. Eclipse, a software tool for developing Java programs, was also installed on the computer to develop the NVM-based encrypted information retrieval system, as shown in Fig. 5.

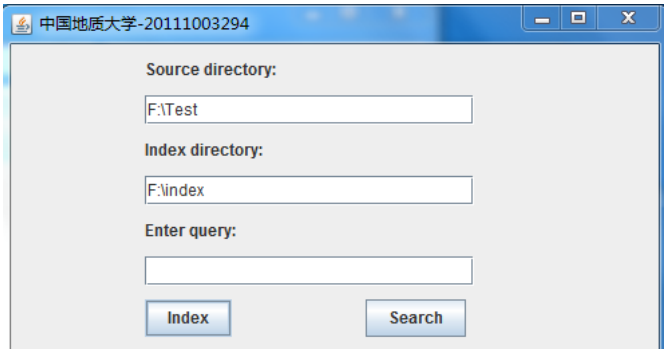

Fig. 5 Screen shot of the encrypted information retrieval system

4.2.1 Effect of document size

The response time is one of the technical indicators for evaluating information retrieval systems. Evaluation criteria include two aspects, namely, the waiting time to enter the search engine and the time to wait for the query results. Web search is normally much faster than manual search, but the user needs to withstand the long waiting time on the Internet. In addition, if the waiting time was too long, the information retrieval system would stop searching over the network, which causes the user not able to link to the web search results. So the user has to abandon the information retrieval system with an unreasonable long response time, no matter how strong are other functions. Thus, the response time plays an important role in evaluating the performance of the information retrieval system.

Table 1 Index building test results

File type	Document size (Byte)	Index building time (ms)
Txt	14112	968
Txt	161159	1434
Doc	12283	866
Doc	125435	1199
Doc	161275	1874
Pdf	95976	1127
Pdf	131761	1337

In the experiments, the index building time was measured and used to verify the effectiveness of the NVM-based encrypted information retrieval system. Table 1 shows the results of the index building tests, including file type, document size, and index building time. The average document size was 100286 bytes, and the average index building time was measured to be 1258 milliseconds (ms) using the proposed encrypted information retrieval system.

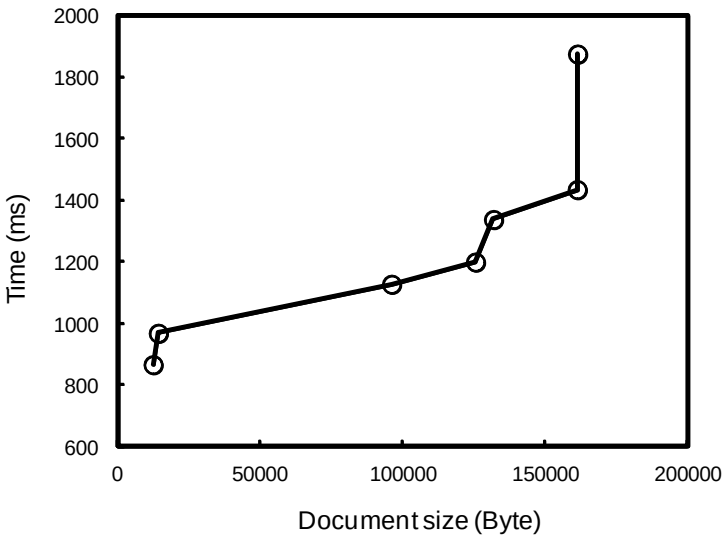

Fig. 6 Effect of document size on the index building time

Figure 6 describes the changes in the time of building the index using the proposed information retrieval system with the size of the document being searched. As the graph shows, the index building time increased with the increase in the content of the document being searched. These results suggest that the more entries are in the document, the more operations are needed to establish the index, that is, taking longer to build the index required by the proposed encrypted information retrieval system.

4.2.2 Effect of encrypted multimedia content

To test encrypted multimedia networking data in this study, the Cranfield data collection containing 1400 documents and 225 queries was used as the file source to be searched to verify the effectiveness of the proposed information retrieval system. As the size of the data collection was comparable to the size of electronic mails, the data collection was appropriate to be used as the test collection. The documents in the collection were encrypted by symmetric encryption schemes or public key encryption schemes [24]. Having the encrypted documents being retrieved, the rankings of the retrieved documents containing one or more of keywords in the query were then computed using equation (10).

Table 2 shows the results of the index building tests using the proposed information retrieval system at various multimedia content levels. According to statistical analysis, the average document size was 38242 bytes, the average multimedia content size was 23962 bytes, and the average index building time was measured to be 1665 milliseconds (ms). Close analysis of the data listed in Table 2 shows that, the index building time increased significantly with the increasing multimedia content size, it is possible that the patterns observed may be the result of multimedia content hindering the search process.

Table 2 Index building times vs. multimedia content levels

Document size (Bytes)	Multimedia content size (Bytes)	Index building time (ms)
18012	5579	1482
20007	7503	1591
36200	21562	1669
38172	23511	1513
39503	24467	1607
44356	29293	1747
71446	55821	2044

Figure 7 shows the effect of encrypted multimedia content level, the ratio of the multimedia content size to the document size, on the index building time. It can be seen in the figure that, there were fluctuations in the index building time when the multimedia content ratio was around 0.6, but the overall trend was upbeat. The index building time increased with the multimedia content ratio with a sharp rise between 0.6 and 0.8, which would suggest a great impact of the multimedia content on the index building time, in turn, the performance of information retrieval.

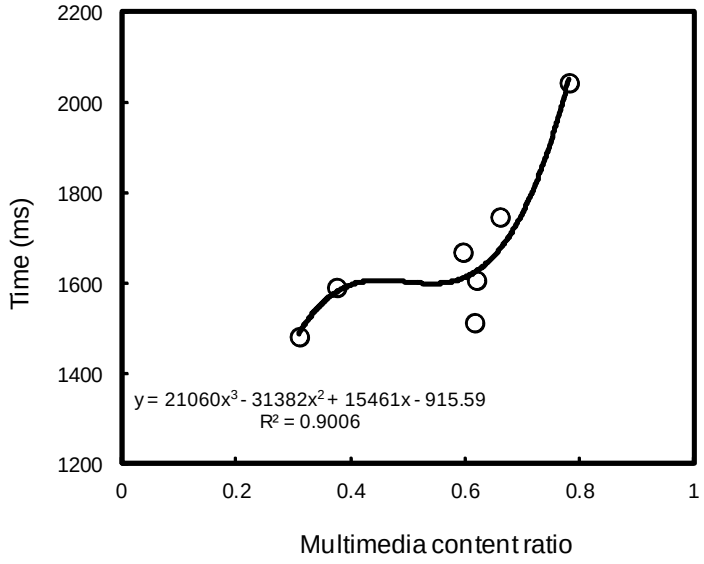

Fig. 7 Effect of encrypted multimedia content on the index building time

By simulating the experimental results, the following polynomial expression between the index building time in ms (y) and the multimedia content ratio (x) was derived:

$$y = 21060x^3 - 31382x^2 + 15461x - 915.59 \quad \text{with} \quad R^2 = 0.9006 \quad .$$

4.2.3 Performance comparisons with other related methods

To evaluate the proposed NVM-based encrypted information retrieval system, two evaluation criteria were used in the experiments to verify the effectiveness of the proposed retrieval system. The retrieval recall and the retrieval precision are two important indicators of the performance of information retrieval from a large document collection.

The retrieval recall is defined as the ratio of the number of the documents retrieved from the search engine to the total number of all relevant documents in the collection, as shown in equation (11). Since information in the network environment is in a dynamic process (e.g. updating, deleting, adding, etc), it is not easy to measure the retrieval recall. An acceptable practice is to estimate the relative retrieval recall of the search engine as a

supplement to the retrieval recall. The information relative recall is subject to manoeuvrable, and thus human uncertainty factors.

$$Recall = Number\ of\ retrieved\ relevant\ documents / Number\ of\ all\ relevant\ documents \quad (11)$$

The retrieval recall depends on the number of the documents that the keyword points to in the index and the number of the documents actually containing the keyword. When a keyword is indexed, the documents containing the keyword are worded out, so the index of the keyword contains all of these documents; thus, the retrieval recall depends on word segmentation.

The precision of retrieval system is defined as the ratio of the total number of retrieved relevant documents to the total number of all retrieved documents, as shown in equation (12):

$$Precision = Number\ of\ retrieved\ relevant\ documents / Number\ of\ all\ retrieved\ documents \quad (12)$$

The retrieval recall and the retrieval precision are contradictory. Upon reaching a certain threshold, the increase of the recall would necessarily cause the precision to reduce; the improvement of precision would also lead to a low recall. In the evaluation of search engine, the requirements of the user's services should be taken into account as the standard; the users tend to achieve a high retrieval precision. Regardless of the emphasis on precision and/or recall, new and important relevant contents on the network should not be ignored. The retrieval precision depends on the total number of the documents, pointed by the keyword to be searched, that actually contain the keyword in the index, and the total number of the documents pointed by the search engine [13].

To compare the proposed N-Level vector model-based relevancy ranking scheme with other related schemes in terms of retrieval recall and precision, we constructed a database to store a massive collection of documents, which was a combination of the Cranfield data collection containing 1400 documents and 225 queries, and a collection consisting of 9790 webpage documents downloaded from the Internet using the revised 'MapReduce' SOM crawler [25].

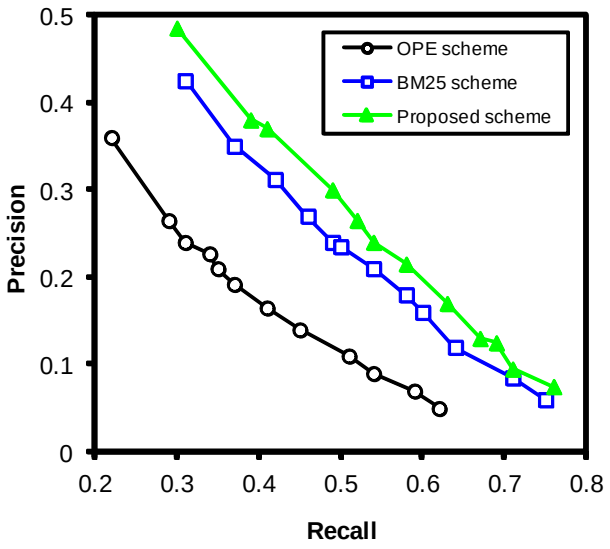

Fig. 8 Comparisons in precision and recall between the proposed scheme with other schemes

Fig. 8 shows the precision and recall (P-R) curves of the proposed N-Level vector model-based relevancy ranking scheme (Proposed scheme), the order preserving encryption-based scheme (OPE scheme) [26], and the BM25 scheme [24]. The multiple-line graph demonstrates that the retrieval precision decreased gradually with the increase in the retrieval recall for all the three schemes, which is in agreement with the contradictory between precision and recall as expected.

Comparisons in the P-R curves show that the OPE scheme lagged far behind the BM25 scheme and the proposed scheme, that is, the precision and recall with the OPE scheme were both lower than the other two schemes. The less effectiveness of the OPE scheme is probably due to the fact that only the term frequency is considered in the order preserving encryption scheme. The proposed N-Level vector model-based relevancy ranking scheme showed great retrieval precision and large recall ratio in comparison with the OPE scheme and the BM25 scheme, indicating that the proposed scheme-based encrypted information retrieval system was effective in ranking the encrypted documents transmitted over multimedia networks. The advantage of the

proposed scheme lies in the introduction of a new formula of the term weighting, taking into account the location of the feature term in the document to describe the content of the document properly.

5. Conclusion

In this study, we suggested an N-level vector model, taking into account the location of the feature term in the document being searched, to address the problems of the order in which the terms appear in the document being lost in the vector space representation with intuitive weighting when using Vector space model for mass information retrieval.

The proposed N-level vector mode-based relevancy ranking scheme was used to retrieve encrypted information from a collection of multimedia files over networks. As the experimental results shown, the index building time increased with the increase in both the document size and the multimedia content of the document to be searched, which suggest that the more multimedia entries are in the document, the more operations are needed to establish the index, that is, taking longer to build the index required by the proposed information retrieval system. Performance evaluation indicated that the proposed NVM-based encrypted information retrieval system is effective in ranking the encrypted documents transmitted over multimedia networks, with large recall ratio and great retrieval precision.

Further experiments should investigate into how fully homomorphic encryption could be applied to information retrieval of encrypted multimedia networking data so as to enhance the retrieval performance.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61272469 and Grant 61303237, and the Wuhan Scientific Research Program under Grant 2013010501010144.

The authors would like to thank anonymous reviewers for their valuable suggestions.

References

[1] Abbadi I (2014) Cloud Management and Security. Wiley, West Sussex

[2] Weis J, Alves-Foss J (2011) Securing Database as a Service: Issues and Compromises. Security Privacy, IEEE 9(6):49-55

[3] Buyya R, Teo C, Venugopal S (2008) Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. Proceedings of 10th IEEE International Conference on High Performance Computing and Communications (HPCC'08) 2008:5-13

[4] Goldreich O, Ostrovsky R (1996) Software protection and simulation on oblivious RAMs. Journal of the ACM 43(3):431-473

[5] Yerukhimovich A (2015) A General framework for one database private information retrieval. http://www.cs.umd.edu/Grad/scholarlypapers/papers/Arkady-pircomp.pdf. Accessed 2 March 2015

[6] Song D, Wagner D, Perrig A (2000) Practical techniques for searches on encrypted data. Proceedings of 2000 IEEE Symposium on Security and Privacy (S&P 2000) 2000:44-55

[7] Boneh D, Di Crecenzo G, Ostrovsky R, Persiano G (2004) Public key encryption with keyword search. Advances in Cryptology-Eurocrypt 2004. Springer. 2004:506-522

[8] Golle P, Staddon J, Waters B (2004) Secure conjunctive keyword search over encrypted data. Proceedings of Applied Cryptography and Network Security. Springer. 2004:31-45

[9] Abdalla M, Bellare M, Catalano D et al (2005) Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. Proceedings of Advances in Cryptology-CRYPTO 2005. Springer. 2005:205-222

[10] Chang Y, Mitzenmacher M (2005) Privacy preserving keyword searches on remote encrypted data. Proceedings of Applied Cryptography and Network Security. Springer. 2005:391-421

[11] Goh EJ (2003) Secure indexes. An early version of this paper first appeared on the Cryptology ePrint. Archived on 7 October 2003

[12] Park DJ, Kim K, Lee PJ (2005) Public key encryption with conjunctive field keyword search. Proceedings of the 2004 Workshop on Information Security Applications. 2005:73-86

[13] Baeza-Yates R, Ribeiro-Neto B et al (1999) Modern information retrieval, volume 82. Addison-Wesley New York

[14] Salton G, Wong A, Yang CS (1975) A vector space model for automatic indexing. Communication of the ACM 18(11):613-620

[15] Salton, G, Mc Gill, MJ (1983) Introduction to Modern Information Retrieval. McGraw-Hill, New York

[16] Baeza-Yates R, Ribeiro-Neto B (1999) Modern Information Retrieval. Addison-Wesley, Harlow, UK

[17] Salton G (1971) The SMART retrieval system-experiments in automatic document processing. Prentice Hall, USA, pp 115-411

[18] Grossman DA, Frieder O (1998) Information retrieval: algorithms and heuristics. Kluwer Academic Publishers, Boston, MA

[19] Manning CD, Schutze H et al (1999) Foundations of statistical natural Language processing. MIT Press, Cambridge

[20] Gao J, Nie J Y, Zhang J et al (2001) TREC-9 CLIR experiments at MSRCN, NIST Special Publication

[21] Lyubashevsky V, Peikert C, Regev O (2010) On ideal lattices and learning with errors over rings. Advances in Cryptology-EUROCRYPT 2010:1-23

[22] Brakerski Z, Vaikuntanathan V (2011) Efficient fully homomorphic encryption from (standard) LWE.

Proceedings of IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS2011), Palm Springs, CA, USA, 2011:97-106

[23] Goldwasser S, Kalai Y, Peikert C et al (2010) Robustness of the learning with errors assumption. Proceedings of Innovations in Computer Science (ICS) 2010:230-240

[24] Manning CD, Raghavan P, Schutze H, Corporation E (2008) Introduction to information retrieval. vol. 1. Cambridge University Press, Cambridge, UK

[25] Ahmed S, Pan P, Tang S (2010) Clustering websites using a MapReduce programming model. J Communication Computer 7:18-26

[26] Agrawal R, Kiernan J, Srikant R, Xu Y (2004) Order preserving encryption for numeric data. Proceedings of the 2004 ACM SIGMOD international conference on Management of data. ACM; 2004:563-574